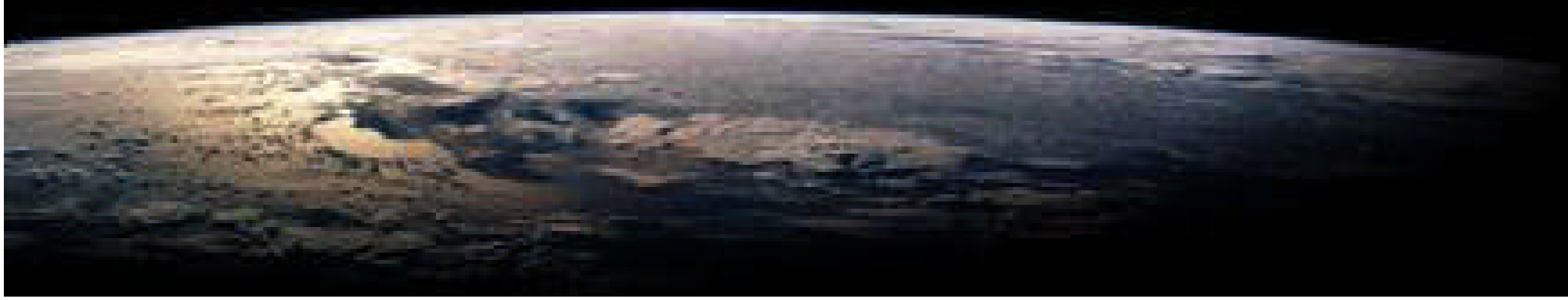
Software Past, Present, and Future: View from the NASA CIO

NASA Software Engineering Workshop

December 2, 1999

Lee Holcomb



Software Past

- High-level language evolution (Fortran, Ada, C/C^{++}, Java) … higher productivity, lower confidence
- Development and use of CMM
- Limited success of software reuse (NetLib)
- No silver bullet
- Hardware capacity (Moore’s Law) outstrips software productivity
- Internet software development process (90-day time box)

Software Present

- Software development costs exceed plans and deliveries continue to be late
  - Costs often exceed plan by 50%, sometime by 100%
  - Most missions have a major software problem
  - Software intense projects are often 2 years late
- Software processes are still chaotic
- Software managers are not well trained
- Still no silver bullet
- Turnover of IT professionals is high

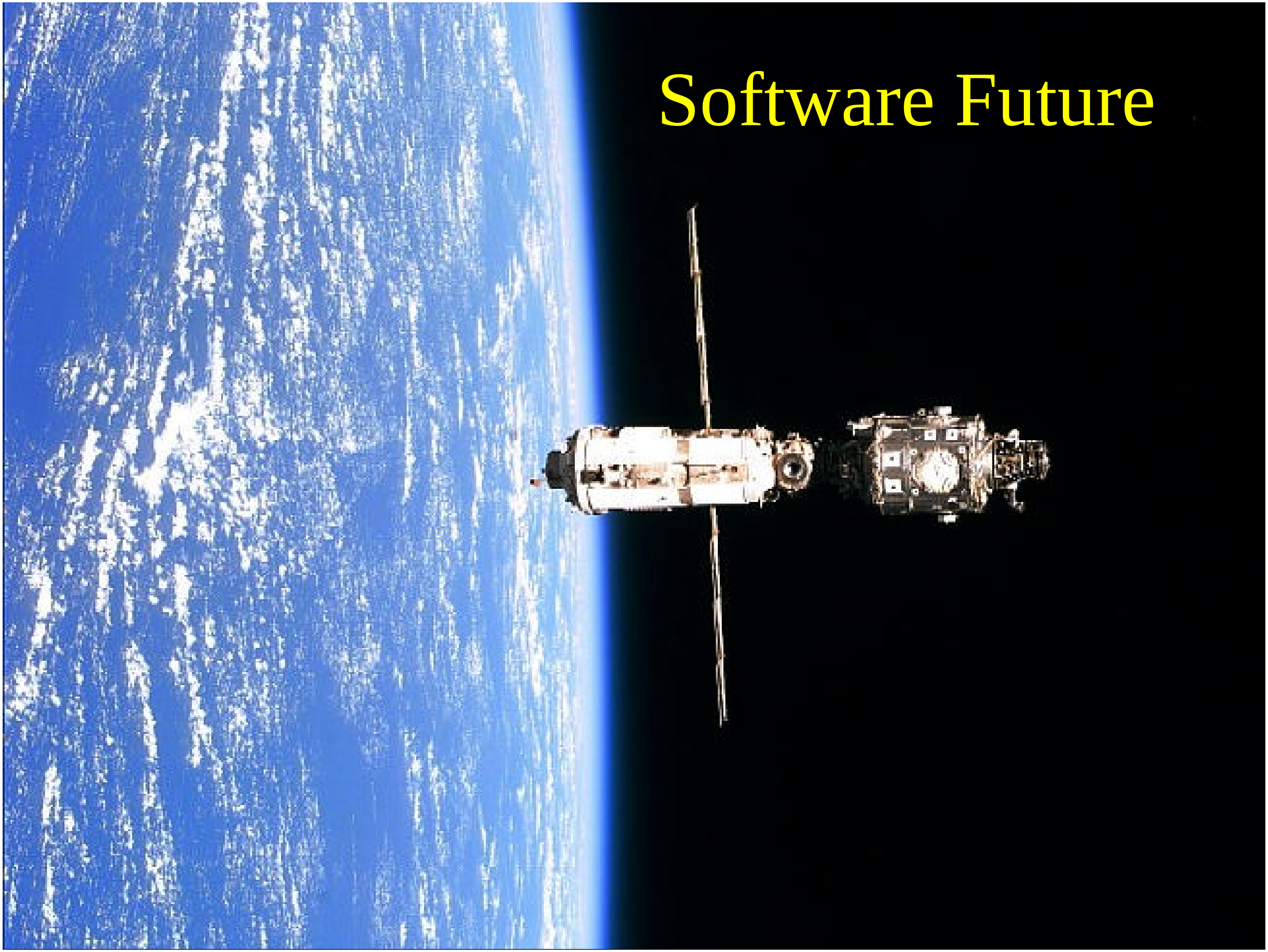
NASA’s Largest Software Challenges

- Earth Observing System Data and Information System
  - NASA design, contractor developed, > Million Lines of Code (MLOC), COTS components
- Checkout Launch Control System
  - NASA design and development, > MLOC, COTS components
- Integrated Financial Management System
  - Contractor provided COTS >MLOC product

8330 Software Projects in Industry
Standish Group's 1994 Report

- 16 % were successful
  - In budget
  - On time
  - Met requirements
  - For large projects, only 9% were successful
- 53 % were "challenged"
  - Average 189% over budget
  - 222% late
  - 39% capabilities missing
- 31 % canceled during development

Software Future



- COTS
  - Market cycle yields poorly-tested, high-risk software
  - Complex software projects planned as all COTS evolve into COTS plus custom developed software
  - Customers with high-confidence applications will demand quality COTS
- Reuse/Formal Methods
  - Software reuse and formal methods have strong potential to improve quality and reduce cost
  - Reuse is still limited to well defined narrow functions
  - Formal methods have been limited to computer hardware or simple software applications
- Open source movement
  - Offers potential for thoroughly examined modular code
- Software development becomes a science

CMM Model : SEI Levels

NOW ➡ 1) Initial: Software process ad hoc, chaotic. Success depends on heroics.

2) Repeatable: Processes established to track cost, schedule, functionality

3) Defined: Process for management and engineering activities documented, standardized, and integrated

4) Managed:Detailed measures of software process and product quality collected

5) Optimizing: Continuous improvement

System Engineering Quality
Also Part of the Problem

- Most projects are now software intense
  - All modern system developments involve software
  - 90% of functionally is provided by software
- System engineering is the work above the software engineering layer
  - Requirements, architecture, risk management, integration, system testing, validation
- Quality system engineering is a prerequisite to quality software engineering
  - Must be partitioned into manageable elements
  - System engineers often have little software expertise

University Environment Trends

Will Increase the Problem in Software Engineering

- Undergraduate
  - Demand for graduates in computer science continues to exceed the supply of graduates
  - High starting salaries are increasing rate of dropouts
- Advanced computer science degrees
  - At one leading university computer science applicants dropped from 300 per year to 20 per year
  - Faculty members are being drawn into industry reducing the ability to train students
- Academic computer science research is declining

NASA Software Engineering Goals

1. Implement software engineering processes that are certified to Level 3 on the CMM scale for all NASA centers
   - Achieve level 3 in three years at 3 centers
2. Conduct software research to enable the development of large trusted software systems
4. Develop with universities a core curriculum for training software managers, software engineers, practitioners, and assurance personnel
5. Define and implement meaningful metrics